Pertemuan 6

Method, Class dan Object pada Java

Objektif :

1. Mahasiswa dapat memahami perbedaan method, class dan objek dalam Java
2. Mahasiswa dapat melihat bentuk umum dari method, class, dan objek dalam Java
3. Mahasiswa dapat memahami cara pemanggilan dan pemberian parameter ke dalam method
4. Mahasiswa dapat membuat program sederhana menggunakan method pada Java
5. Mahasiswa dapat membuat program sederhana menggunakan class dan objek pada Java

P6.1 Teori

1. Method pada Java

Metode (*Method*) adalah sekumpulan statement program yang disatukan menjadi sebuah subprogram atau fungsi, diawali dengan tanda "{" diakhiri dengan tanda "}". Ada 2 macam metode dan 1 metode pengendali, yaitu:

- Metode kelas : Metode ini dapat dieksekusi walaupun tidak terdapat objek dalam kelas tersebut. Seperti variabel kelas, metode kelas juga dideklarasikan menggunkan keyword static.
- Metode objek : Metode ini hanya dapat dieksekusi sehubungan dengan objek tertentu.
- Metode main() : Metode ini digunakan pada saat aplikasi Java dimulai, menggunakan keyword static. Sebelum aplikasi mulai dieksekusi, diperlukan metode walaupun tanpa objek.

Metode adalah suatu blok dari program yang berisi kode dengan nama dan properti yang dapat digunakan kembali. Metode dapat mempunyai nilai balik atau tidak, penjelasan beserta contohnya adalah sebagai berikut:

<u>Metode tidak membalikkan nilai</u>

Jika diberi awalan dengan kata void maka metode tersebut tidak memberi nilai balik.

contoh:

```
void Namametode(){
System.out.println("INI METODE");
}
```

<u>Metode membalikkan nilai</u>

Jenis kedua adalah jika metode diberi awalan sebuah tipe data maka metode tersebut akan memberi nilai balik data yang bertipedata sama dengan metode tersebut.

contoh :

```
int Namametode(){ int nilai;
System.out.println("Kasih nilai balik");
return nilai; // mengembalikan suatu nilai dari metode
}
```

Selain dua jenis di atas metode juga ada yang diberi parameter

contoh :

```
void Namametode(String a){
System.out.println("INI METODE"); }
```

Karakteristik Method

Berikut adalah karakteristik dari method :

1. Dapat mengembalikan satu nilai atau tidak sama sekali
2. Dapat diterima beberapa parameter yang dibutuhkan atau tidak ada parameter sama sekali.
3. Setelah method telah selesai dieksekusi, dia akan kembali pada method yang memanggilnya.

Memanggil Instance dan memberikan Variabel dari Method

Untuk mengilustrasikan bagaimana memanggil method, mari kita menggunakan class string sebagai contoh. Anda dapat menggunakan the dokumentasi dari Java API untuk melihat semua method yang tersedia dalam class string. Selanjutnya, kita akan membuat method, kita sendiri. Tapi untuk saat ini, mari terlebih dahulu kita gunakan method yang sudah disediakan oleh Java.

Untuk memanggil sebuah instance method, kita dapat menuliskan :

```
nameOfObject.nameOfMethod( parameters );
```

mari kita mengambil dua contoh method yang ditemukan dalam class String.

Deklarasi method	Definisi
public char charAt(int index)	Mengambil karakter pada indeks tertentu.
public boolean equalsIgnoreCase (String anotherString)	Membandingkan antar String, tidak case sensitive.

Contoh program seperti di bawah ini :

```
String str1 = "Hello";
char x = str2.charAt(0);
String str2 = "hello";
boolean result = str1.equalsIgnoreCase( str1 );
```

Pemberian Variabel dalam Method

Pada contoh kita sebelumnya , kita sudah pernah mecoba melewati variabel pada method. Walaupun kita belum dapat membedakan antara perbedaan tipe variabel yang diberikan (*passing*) ke method dalam Java. Ada dua tipe data variabel *passing* pada method, yang pertama adalah pass-by-value dan yang kedua adalah pass-by-reference.

Pass-by-Value

Ketika pass-by-values terjadi, method membuat sebuah salinan dari nilai variable yang dikirimkan ke method. Walaupun demikian, method tidak dapat secara langsung memodifikasi nilai variabel pengirimnya meskipun parameter salinannya sudah dimodifikasi
nilainya di dalam method.

Contoh :

```
public class TestPassByValue
{
public static void main( String[] args ){
int i = 10;
//mencetak nilai i
System.out.println( i );
//memanggil method test
//passing i pada method test test( i );
//Mencetak nilai i
System.out.println( i );
}
public static void test( int j ){ //merubah nilai parameter j
j = 33;
}
}
```

Pada contoh di atas, kita memanggil method tes dan melewatkan nilai variabel i sebagai parameter. Nilai pada i disalinkan ke variable j pada method. Pada kondisi ini variabel jadalah merupakan variabel pengganti pada method tes, jika nilai j berubah maka

nilai variabel i yang terletak pada main tidak akan ikut berubah walaupun awalnya variabel j merupakan salinan dari variabel i.

Pass-by-reference

Ketika sebuah pass-by-reference terjadi, alamat memori dari nilai pada sebuah variable dilewatkan pada saat pemanggilan method. Hal ini berarti bahwa method menyalin alamat memori dari variabel yang dilewatkan pada method. Ini tidak seperti pada pass-by-value, method dapat memodifikasi variabel asli dengan menggunakan alamat memori tersebut, meskipun berbeda nama variabel yang digunakan dalam method dengan variabel aslinya, kedua variabel ini menunjukkan lokasi dari data yang sama.

contoh :

```
class TestPassByReference
{
public static void main( String[] args ){
//membuat array integer
int []ages = {10, 11, 12};
//mencetak nilai array
for( int i=0; i<ages.length; i++ ){
System.out.println( ages[i] );
}
}
test( ages );
for( int i=0; i<ages.length; i++ ){
System.out.println( ages[i] );
}
public static void test( int[] arr ){ //merubah nilai array
for( int i=0; i<arr.length; i++ ){
arr[i] = i + 50;
}
}
}
```

Memanggil Method Static

Method Static adalah method yang dapat dipakai tanpa harus menginisialisasi suatu class (maksudnya tanpa menggunakan variabel terlebih dahulu). Method static hanya dimiliki oleh class dan tidak dapat digunakan oleh instance (atau objek) dari suatu class. Method static dibedakan dari method yang dapat instance di dalam suatu class oleh kata kunci static.

Untuk memanggil method static, ketik :

Classname.staticMethodName(params);

Contoh dari static method yang digunakan :

System.out.println(“Hello world”);

int i = **Integer.parseInt**(“10”);

String hexEquivalent = **Integer.toHexString**(10);

Konstruktor

Konstruktor adalah suatu metode yang dapat digunakan untuk memberi nilai awal pada saat objek diciptakan. Konstruktor akan dipanggil secara otomatis begitu objek diciptakan. Konstruktor memiliki ciri :

a. namanya sama dengan nama kelas

b. Tidak mengembalikan nilai (dan juga tidak boleh ada kata void didepannya)

Jika constructor tidak didefinisikan, Java memberikan constructor dengan nama constructor_default. Constructor default tidak melakukan apa-apa, namun semua variabel yang diinisialisasi dianggap sebagai berikut:

- Variabel numerik diset ke 0
- String diset ke null
- Variabel boolean di set ke false

Constructor tidak memiliki tipe hasil, walaupun constructor bisa public, private, atau protected. Sebagian constructor bersifat public.

contoh :

```
class Coba
{
Coba(){ //Ini yang namanya konstruktor
```

```
System.out.println("Ini Konstruktor");
}
public static void main(String[] args)
{ Coba obj=new Coba();
}
}
```

Jika konstruktor dipanggil dari kelas turunan, maka caranya adalah dengan menuliskan kata super(); pada class turunan. Untuk penurunan sifat akan dibahas pada pertemuan selanjutnya Konstruktor juga ada yang diberi parameter

contoh :

```
konstrk(String a){
System.out.println("INI KONSTRUKTOR");}
```

2. Class dan Objek pada Java

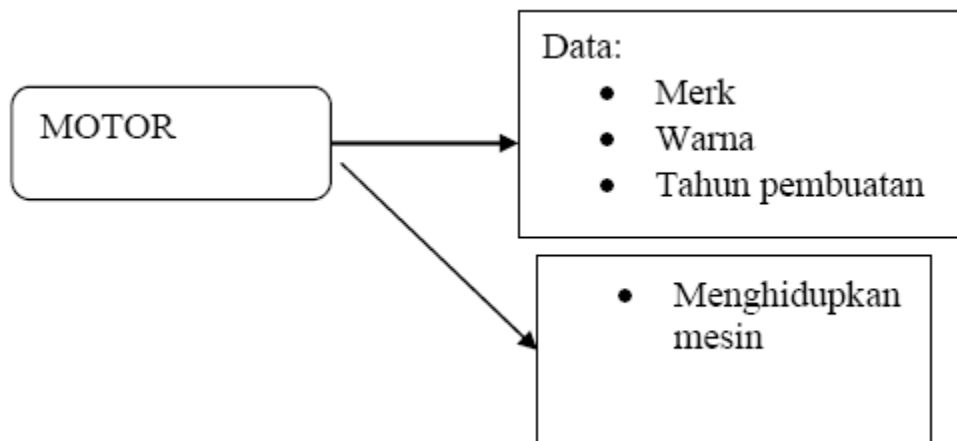
Kelas (*class*) merupakan salah satu konsep fundamental pemrograman berorientasi objek. Kelas dapat diilustrasikan sebagai suatu cetak biru (blue print) atau prototipe yang digunakn untuk menciptakan objek.

Definisi kelas terdiri atas dua komponen, yaitu deklarasi kelas dan body kelas. Deklarasi kelas adalah baris pertama di suatu kelas, dan minimal mendeklarasikan nama kelas. Sementara itu, body dideklarasikan setelah nama kelas dan berada diantara kurung kurawal.

```
//deklarasi kelas
Public class ContohKelas {
//body kelas
}
```

Pada Java, nama kelas sekaligus merepresentasikan nama file kode program dan sifatnya ***case sensitive***.

Objek adalah entitas dasar saat runtime. Pada saat kode program dieksekusi, objek berinteraksi satu sama lain tanpa harus mengetahui detil data atau kodenya. Interaksi antara objek ini dilakukan menggunakan suatu message.

Pada pemrograman berbasis objek, objek dijadikan sebagai komponen utama dalam program, objek menggabungkan data dan fungsi sebagai satu kesatuan. Dalam pemrograman berbasis objek terdapat dua istilah yang sangat terkenal yaitu class dan objek. Pengertiannya adalah sebagai berikut class adalah cetak biru dari sebuah objek, jadi kita dapat membuat banyak objek dari sebuah class, atau kita dapat analogikan, class itu adalah cetakan puding, sedangkan objek adalah puding. Contoh sebuah class adalah motor. Class motor memiliki data merk, warna, tahun pembuatan dan juga memiliki metode seperti menghidupkan mesin, kecepatan dsb.

Jadi jika ada motor Bowo, motor Raga dan motor Loan maka itu adalah sebuah objek dari class motor. Pada pemrograman java, cara untuk menciptakan sebuah objek dari suatu class adalah dengan cara sebagai berikut :

<nama class> <nama objek>=new <nama konstruktor>

Misal:

```
String str = new String();
Random r = new Random();
Pegawai p2 = new Pegawai();
Date hari = new Date();
```

Hari adalah object reference dari class Date yang akan digunakan untuk mengakses class Date. Sedangkan operator *new* adalah operator yang akan menghasilkan hari sebagai reference ke instance dari class Date ().

Contoh :

```
class motor
{
int warna;
String merk;
void hidupkanMesin(){
System.out.println("Ini Metode hidup");
}
public static void main(String[] args)
{
motor MotorBowo=new motor(); //kelas dibuat
MotorBowo.warna="Hitam"; //memakai data warna
System.out.println(MotorBowo.warna);
MotorBowo.hidupkanMesin(); //memanggil metode}}
```

Instansiasi Class

Untuk membuat sebuah objek atau sebuah instance pada sebuah class. Kita menggunakan operator **new**. Sebagai contoh, jika anda ingin membuat instance dari class string, kita menggunakan kode berikut :

```
String str2 = new String("Hello world!");
```

Ini juga sama dengan,

```
String str2 = "Hello";
```

Menentukan Class dari sebuah Object

Jika kita ingin mengetahui class dari sebuah obyek dapat dilakukan dengan cara :

1. **Method getClass()** mengembalikan sebuah obyek Class (dimana Class itu sendiri merupakan sebuah class) yang memilki sebuah method getName(). Selanjutnya getName() akan mengembalikan sebuah string yang mewakili nama class.

Sebagai contoh,

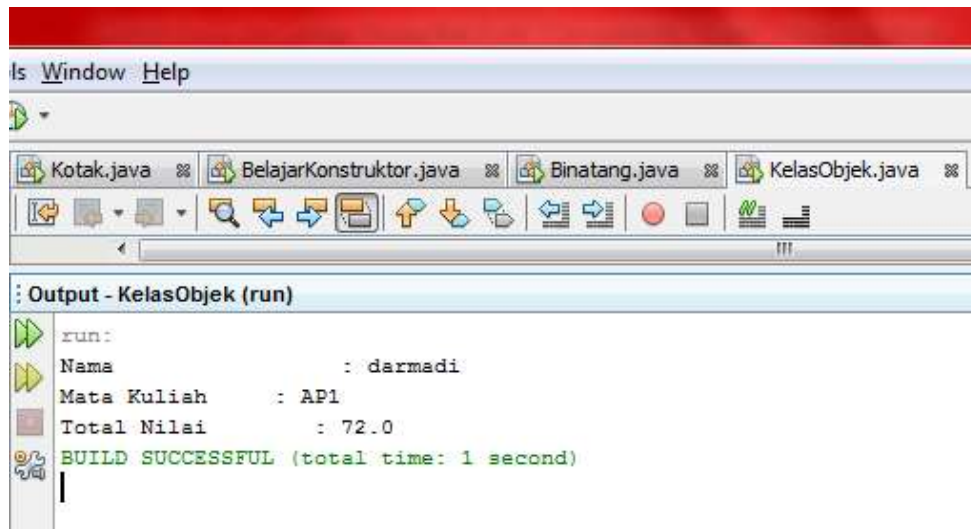
String name = key.getClass().getName();

2. Operator InstanceOf

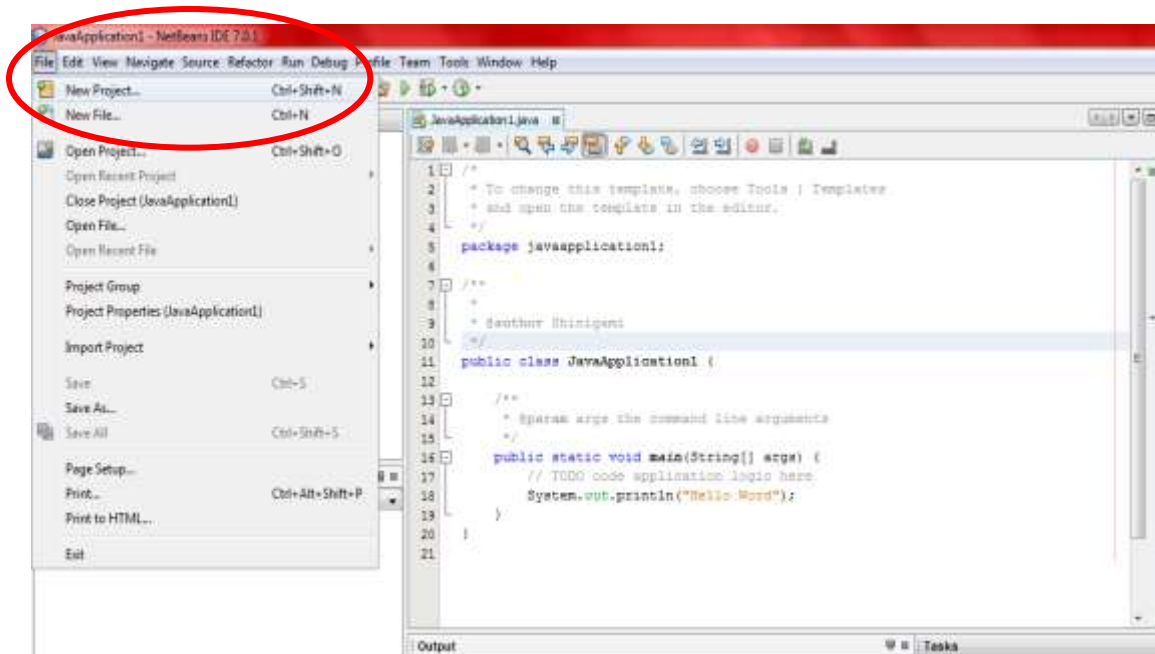
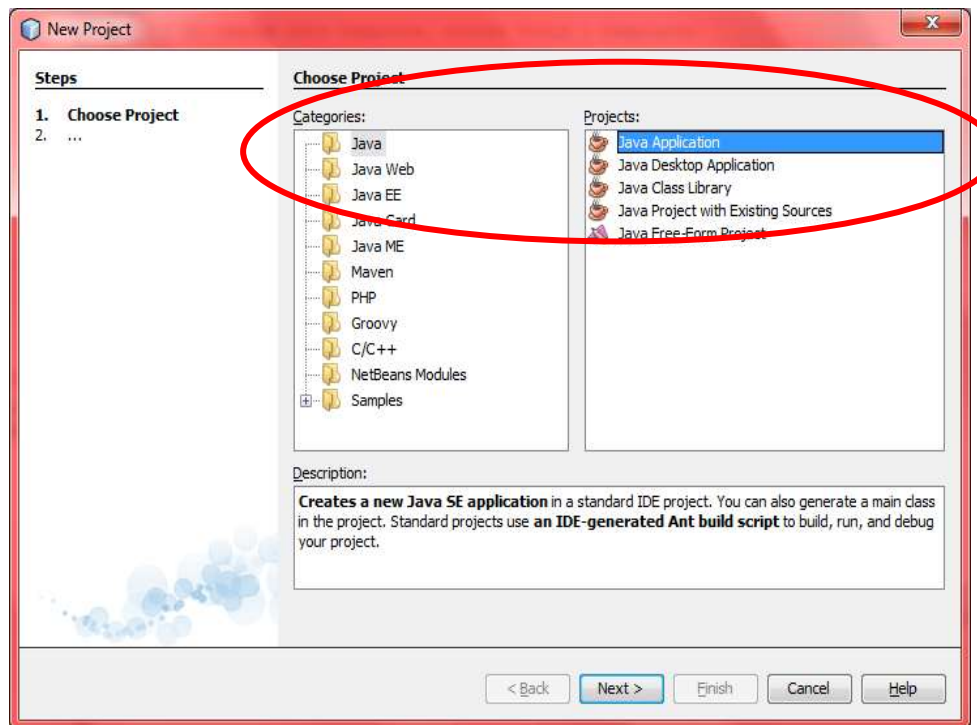
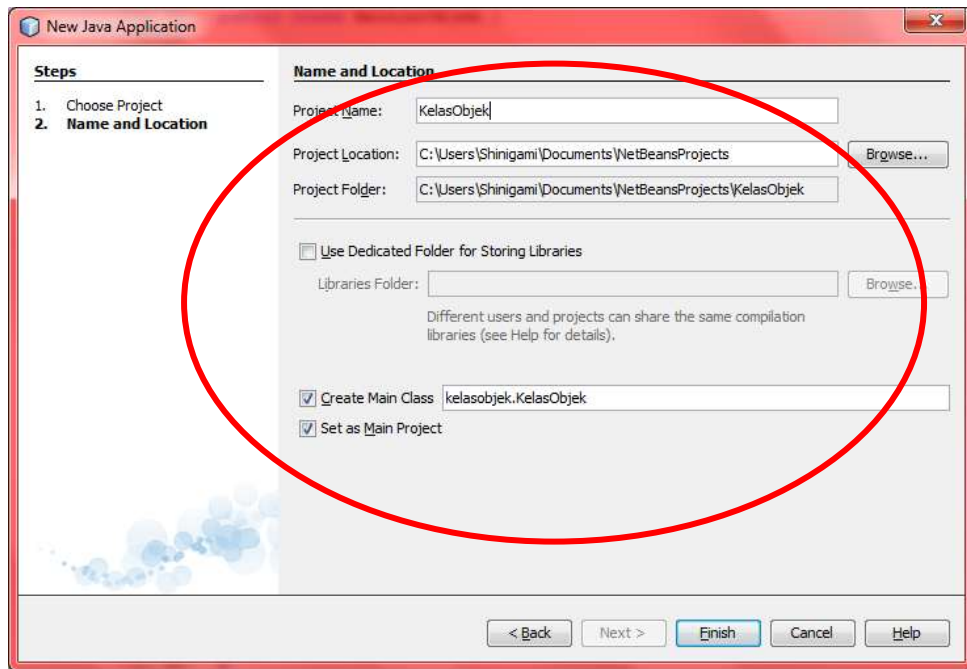
InstanceOf memiliki dua operand: obyek pada sebelah kiri dan nama class pada sebelah kanan. Pernyataan ini mengembalikan nilai true atau false tergantung pada benar/salah obyek adalah sebuah instance dari penamaan class atau beberapa subclass milik class tersebut. Sebagai contoh,

```
boolean ex1 = "Texas" instanceof String; // true
Object pt = new Point(10, 10);
boolean ex2 = pt instanceof String; // false
```

P6.2 Contoh Kasus

Buat program sederhana menggunakan kelas, objek, dan method dengan output :

Langkah-langkah Pengerjaan:

1. Jalankan Netbeans

2. Buat file project baru dengan memilih menu File – New Project, atau dengan menggunakan hotkey Ctrl+Shift+N.

3. Pilih jenis project yang akan dibuat (Java – Java Application)

4. Nama Project beserta nama classnya adalah KelasObjek

5. Ketikkan kode program di bawah ini pada code editor

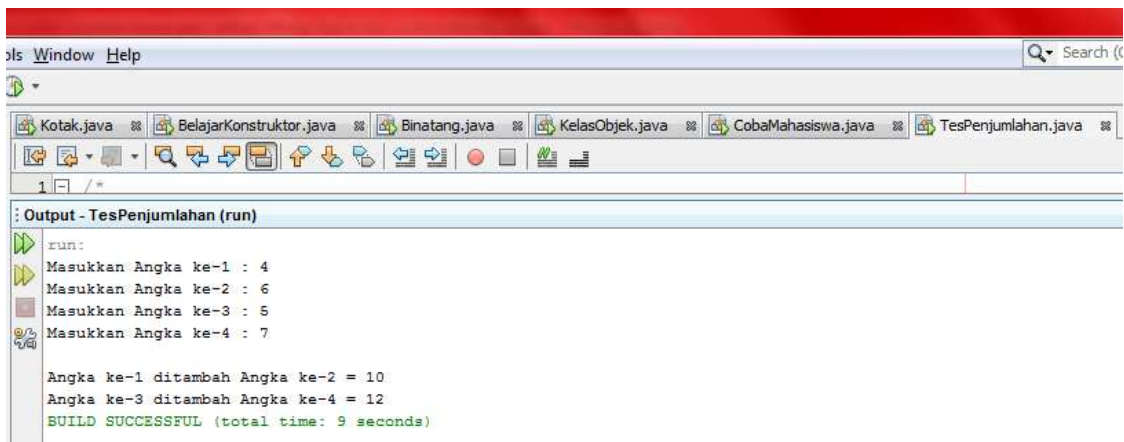
```
package kelasobjek;

public class KelasObjek {

    public static void main(String[] args) {
        // TODO code application logic here
    Hitung Mahasiswa = new Hitung();   // membuat objek Mahasiswa dari kelas Hitung

// memberikan nilai untuk objek Mahasiswa.Nama yaitu darmadi
    Mahasiswa.Nama = "darmadi";

// memberikan nilai untuk objek Mahasiswa.MatKul yaitu AP1
    Mahasiswa.MatKul = "AP1";

// memberikan nilai untuk objek Mahasiswa.uts yaitu 75
    Mahasiswa.uts = 75;

// memberikan nilai untuk objek Mahasiswa.uas yaitu 65
    Mahasiswa.uas = 65;

// membuat objek Skor dari method SkorUjian
    double Skor = Mahasiswa.SkorUjian();

// mencetak nilai dari objek Mahasiswa.Nama
   System.out.println("Nama              : " + Mahasiswa.Nama);

// mencetak nilai dari objek Mahasiswa.MatKul
```

```
        System.out.println("Mata Kuliah     : " + Mahasiswa.MatKul);

    // mencetak nilai dari objek Skor
        System.out.println("Total Nilai       : " + Skor);
            }
        }

    class Hitung
    {
        String Nama, MatKul;
        double uts, uas;
        public double SkorUjian()
            {
            double Total;
            Total = (0.7*uts + 0.3*uas); /* melakukan proses perhitungan dari nilai
                                        * variabel uts dan uas yang dikirim dari kelas utama */
            return(Total);          // mengembalikan nilai ke kelas utama
            }
    }
```

6. Build project tersebut dengan memilih menu Run – Build Main Project, atau dengan menggunakan hotkey F11.

7. Jika tidak ada kesalahan (BUILD SUCCESSFUL), jalankan project tersebut dengan memilh menu Run – Run Main Project, atau dengan menggunakan hotkey F6.

P6.3 Latihan

Buat program sederhana untuk menginput nilai untuk dijumlahkan dan menampilkan hasilnya. Output dari program ini seperti gambar berikut:

Jawaban:

1. Jalankan Netbeans Anda
2. Lakukan langkah-langkah pengerjaan seperti contoh kasus sebelumnya.
3. Pada code editor Netbeans, ketikkan program berikut:

```
package tespenjumlahan;

import java.io.*;
public class TesPenjumlahan {

    public static void main(String[] args) throws Exception{
        // TODO code application logic here
       DataInputStream masuk = new DataInputStream(System.in);

        String strangka1, strangka2, strangka3, strangka4;

        System.out.print("Masukkan Angka ke-1 : ");
        strangka1 = masuk.readLine();
        int angka1 = Integer.parseInt(strangka1);
        System.out.print("Masukkan Angka ke-2 : ");
        strangka2 = masuk.readLine();
        int angka2 = Integer.parseInt(strangka2);
        System.out.print("Masukkan Angka ke-3 : ");
        strangka3 = masuk.readLine();
        int angka3 = Integer.parseInt(strangka3);
        System.out.print("Masukkan Angka ke-4 : ");
        strangka4 = masuk.readLine();
        int angka4 = Integer.parseInt(strangka4);

        System.out.println();
        Penjumlahan tambah1 = new Penjumlahan(angka1,angka2);
        System.out.println("Angka ke-1 ditambah Angka ke-2 = 
"+tambah1.hitungPenjumlahan());

        Penjumlahan tambah2 = new Penjumlahan(angka3,angka4);
        System.out.println("Angka ke-3 ditambah Angka ke-4 = 
"+tambah2.hitungPenjumlahan());
    }
}
class Penjumlahan
{
    int a;
    int b;
    public Penjumlahan(int a, int b)
    {
        this.a = a;
        this.b = b;
    }
    public int hitungPenjumlahan()
```

```
    {
      return a + b;
    }
  }
```

P6.4 Daftar Pustaka

Ady Wicaksono, *Dasar-dasar Pemrograman Java*, PT. Elex Media Komputindo, Jakarta 2002.

Benny Hermawan, *Menguasai Java 2 Object Oriented Programming*, Andi, Yogyakarta, 2004.

Ginanjar Utama, *Berfikir Objek: Cara Efektif Menguasai Java*, 2003.

ANuff, *Penuntun Pemrograman Java*, Andi, Yogyakarta, 1997.

Abdul Kadir, *Dasar Pemrograman Java 2*, Andi Yogyakarta, 2008.